CATALOGAGE DE TRAITEMENTS INFORMATIQUES DU DOMAINE GÉOGRAPHIQUE, CONCEPTION ET EXPLOITATION D'UN MODÈLE DE MÉTADONNÉES

Par Yann ABD-EL-KADER
Institut Géographique National, laboratoire COGIT,
2-4 avenue Pasteur, 94165 SAINT-MANDE Cedex, France
yann.abd-el-kader@ign.fr

Résumé

Dans les organisations telles que l'Institut Géographique National (IGN), les développeurs et les utilisateurs ont besoin d'assistance pour chercher, partager et utiliser les nombreux traitements informatiques existants.
Cet article décrit le travail de la thèse en cours destiné à construire un catalogue des traitements géographiques utilisés à l'IGN. Nous proposons un modèle de métadonnées qui adapte le modèle OWL-S à notre contexte, où les outils ne sont pas nécessairement des services Web. Basée sur le modèle ainsi défini, une application client/serveur a été développée afin de permettre la consultation et l'acquisition de métadonnées.
Nous détaillons en particulier la partie du modèle décrivant les modes d'emploi. Notamment, nous montrons comment calculer dynamiquement leur adaptation au contexte de l'utilisateur , par exemple en fonction des caractéristiques de ses données d'entrées, de celles des données attendues en sortie, de son environnement de travail, ou de ses préférences.

Introduction

L'information géographique – base de données vecteur , base de données image, cartes papiers – est construite, analysée, transformée. Pour cela elle fait l'objet de traitements, communément réalisés par des Systèmes d'Information Géographiques (SIG), bientôt par des services Web. Des organismes développent aussi leurs propres programmes informatiques de traitements géographiques répondant à leurs besoins spécifiques. Par exemple à l'Institut Géographique National (IGN) les équipes de production et les laboratoires de recherche conçoivent et implémentent, entre autres, des traitements d'images, de généralisation [1], d'appariement de base de données. Il s'ensuit que les développeurs et utilisateurs de l'IGN ont besoin d'aide pour partager, rechercher et connaître ces traitements.

Le but de notre travail est de fournir cette aide, qui n'existe actuellement que sous forme de documentations éparses aux formats hétérogènes. Dans cet article nous montrons comment nous avons défini un modèle de métadonnées adapté aux besoins de consultation identifiés, et quelles ont été nos sources d'inspiration. En particulier, nous nous intéressons à la façon de décrire les connaissances d'utilisation des traitements. Afin d'adapter les modes d'emploi au contexte propre à chaque utilisateur (préconditions sur le format des données, disponibilité des traitements annexes et de l'environnement informatique, capacité de l'utilisateur à programmer), nous mettons en œuvre des mécanismes d'inférence. En effet, tous les cas de figure ne peuvent être stockés à l'avance dans la base de métadonnées; il faut donc dériver l'information recherchée des métadonnées présentes. Les règles qui permettent cette dérivation représentent la connaissance des experts, que notre modèle tente de capturer sous une forme opérationnalisable. Ces règles peuvent ainsi être exploitées par l'application que nous avons développée, application qui permet la consultation et la saisie des descriptions de traitements géographiques.

1 Décrire les traitements géographiques

1.1 Les ressources à décrire, les besoins des utilisateurs

Les traitements auxquels nous nous intéressons existent avant tout sous forme informatique (programmes et librairies de fonctions utilisables dans un contexte de développement, logiciels, SIG, plug-in, services Web), et également sous forme non-implémentée (algorithmes). L'analyse des requêtes typiques exprimées par les utilisateurs (*«Où sont disponibles les programmes d'appariement ?», «Est-ce qu'il existe un programme de généralisation adapté aux bâtiments de formes irrégulières ?», «Comment fonctionne le programme Accordéon ?», «Quels sont les avantages de Lamps2 par rapport à Geoconcept ?»* , etc.) montre que pour un traitement donné les besoins d'information portent sur cinq thèmes principaux : les métadonnées qui l'identifient (nom, date, auteur, etc.), «ce qu'il fait», «comment il fonctionne», «comment l'utiliser» et «quelle est son évaluation». L'enquête auprès des utilisateurs a permis de révéler des besoins spécifiques au domaine géographique : par exemple pour comprendre ce que fait un traitement, il est utile de fournir des illustrations graphiques sous forme d'échantillons des données, ainsi qu'une description de l'évolution des propriétés des objets géographiques avant et après traitement. Il est également des besoins qui nécessitent la mise en oeuvre d'un raisonnement «expert» ; nous prenons en section 3 l'exemple des modes d'emploi à adapter au contexte d'utilisation.

[1] Il s'agit de simplifier la représentation des cartes lorsque l'échelle croît, sauvegardant ainsi leur lisibilité pour l'oeil humain à la perception limitée.

1.2 Le modèle de métadonnées

Nous avons dû créer notre propre modèle de métadonnées car dans le domaine de l'information géographique il existe des normes pour décrire les données, mais peu encore pour décrire les traitements. Le comité technique de l'International Organization for Standardization sur l'information géographique et la géomatique [2] propose bien un modèle, l'ISO 19119, mais l'ensemble des descripteurs proposé est trop sommaire pour nos besoins. Finalement c'est dans le domaine du Web sémantique que notre état de l'art des descriptions de traitements nous a mené. C'est en ef fet en nous inspirant d'OWL-S [3], langage de description des services Web, que nous avons défini les principales facettes de description des traitements. On retrouve, à travers le choix des cinq facettes de description adoptées pour notre modèle (fig.1), les trois facettes d'un service selon OWL-S : ServiceProfile (ce que le service fait), ServiceModel (comment fonctionne le service) et ServiceGrounding (comment y accéder) (The OWL Services Coalition, 2004). Source d'inspiration, OWL-S ne répondait néanmoins pas pleinement à nos besoins car, premièrement, seule une partie de nos traitements se présente sous forme de services W eb, les autres possédant des spécificités requérant des descripteurs particuliers; deuxièmement OWL-S est conçu dans un but de planification et d'exécution automatique : les destinataires des descriptions OWL-S sont des agents logiciels. Or, à l'opposé, si nos descriptions de traitements sont formatées pour être l'objet de raisonnement (cf. §3), elles n'en demeurent pas moins au final destinées à la consultation humaine.

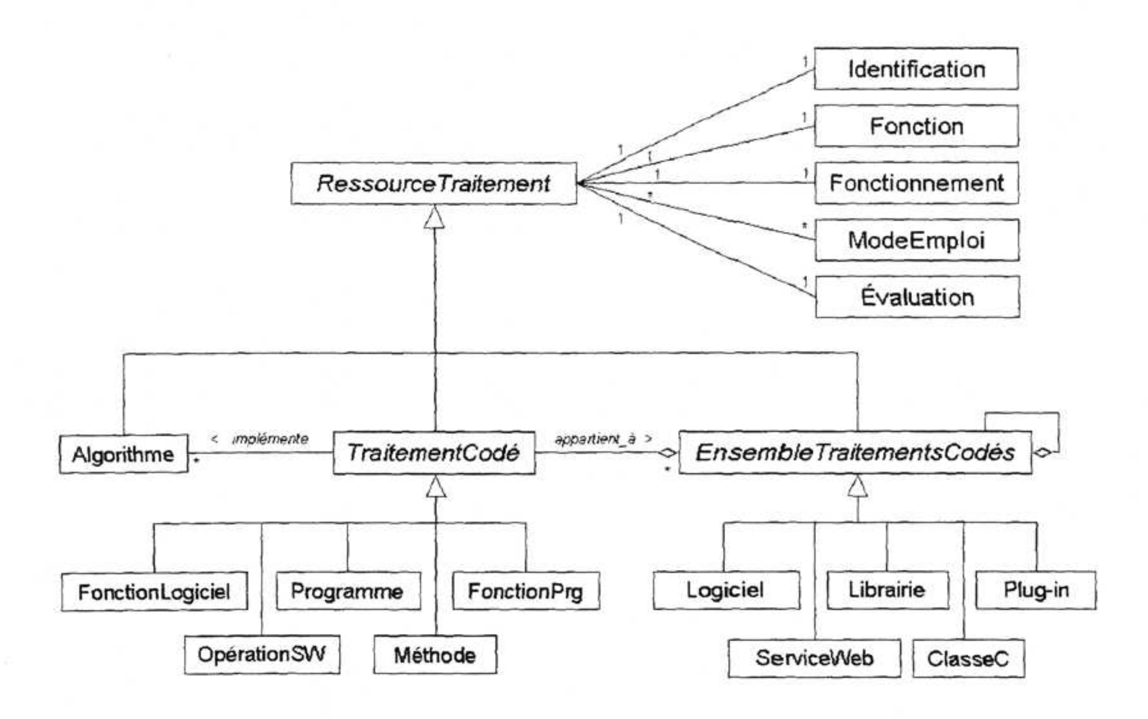


Figure 1-1 : Classes principales de notre modèle de métadonnées

2 Décrire les connaissances d'utilisation

2.1 Les connaissances à représenter

Délimitation de l'objectif

L'utilisation des traitements géographiques nécessite la mobilisation d'ensembles étendus de connaissances. Des connaissances informatiques ou géographiques, contextuelles ou générales, théoriques ou empiriques, explicites ou tacites [4]... que possède l'expert et qui manquent au novice. Notre ambition n'est pas de représenter toutes ces connaissances, mais seulement une partie qui, dans un contexte donné, permettra l'utilisation ef fective du traitement voulu. Pour circonscrire notre cadre de travail, nous commençons par poser des hypothèses sur l'utilisateur . On postule ainsi que les savoir-faire et concepts informatiques de base sont maîtrisés.

Par ailleurs, il n'est pas envisageable de représenter l'intégralité des connaissances nécessaires au paramétrage de certains traitements complexes, ni forcément de fournir toutes les explications nécessaires à une utilisation experte. Par exemple, une partie de la thèse de S.Bard est consacrée au paramétrage des traitements évaluant la qualité des résultats de traitements de généralisation cartographique (Bard, 2003). Nos descriptions ne peuvent se substituer totalement aux manuels, articles et thèses : en raison de ses propriétés propres (taille non limitée, expressivité), le

[2] http://www.isotc211.org/

[3] Ontology Web Language for Services (The OWL Services Coalition, 2004)

[4] La distinction entre connaissances tacites et connaissances explicites a fait l'objet de plusieurs définitions dans le domaine de la gestion de connaissances (knowledge managment). Retenons simplement deux critères : une connaissance est dite explicite si son existence est identifiée, et s'il existe un support quelconque qui en permet la transmission, sinon elle est tacite.

format traditionnel «texte en langue naturelle» reste irremplaçable pour les modes d'emploi des traitements complexes. Les descriptions de notre base de métadonnées ont donc une vocation complémentaire plutôt que concurrente des documentations en langue naturelle. Elles doivent les référencer. En résumé nous souhaitons apporter aux utilisateurs une aide certes limitée en complexité mais néanmoins suffisamment complète pour la majorité des besoins communément rencontrés. Une des principales conditions pour atteindre cet objectif sera la capacité du modèle proposé à permettre l'explicitation des connaissances tacites.

Des modes d'emploi adaptés au contexte d'utilisation

Nous allons être amenés à distinguer plusieurs contextes d'utilisation des traitements, selon que l'utilisateur est prêt à programmer ou non, suivant les contraintes liées aux types de données qu'il utilise, suivant encore l'environnement logiciel dont il dispose ou qu'il est contraint d'utiliser pour des raisons de compatibilité avec d'autres traitements. Ces quelques exemples montrent que l'aide apportée à l'utilisateur ne peut être figée. L'adapter automatiquement en fonction du contexte suppose deux choses. Premièrement, les connaissances générales sur l'environnement des traitements doivent être représentées. Il faut fournir un cadre qui permette à l'expert de les exprimer . Deuxièmement, ces connaissances doivent être formalisées de façon à être exploitables par l'application de gestion des métadonnées.

2.2 Quelques sources d'inspiration

Il existe divers modèles de gestion de connaissances dont on peut s'inspirer pour représenter les modes d'emploi. (Bandza, 2000) a fait une synthèse de quelque uns de ces modèles. En ce qui nous concerne, nous avons notamment étudié ceux proposés par la méthode MASK (Method for Analysing and Structuring Knowledge), dont le but est le recueil et la capitalisation des savoirs tacites d'experts (Bézard et Ariès, 2003). L'ancêtre de MASK est MKSM (Methodology for Knowledge System Management), conçu par J-L.Ermine afin de représenter les connaissances au sein du C.E.A. (Commissariat à l'Energie Atomique). L'élaboration de la méthode a pour origine la difficulté d'acquérir les connaissances tacites des experts parfois difficilement exprimables. D'où la nécessité de fournir un cadre afin de faciliter ce qui est avant tout un problème d'acquisition des connaissances. On trouvera dans (Ermine, 2003) la description des six modèles MASK traduisant des points de vue *Connaissances fondamentales, Activités, Contexte historique, Savoir-faire, Concepts* et *Historique des solutions et leurs justifications*.

Concernant les connaissances heuristiques à représenter pour l'adaptation des traitements (par exemple un expert peut connaître plusieurs façons de contourner un problème d'incompatibilité entre deux programmes, mais ne pas savoir celle qui est préférable), nous nous sommes inspirés des travaux existant dans le domaine de la planification. Par exemple, (Clouard *et al.*, 1998) proposent des descripteurs *déclencheur, précondition, évitement*, etc. pour piloter les applications de traitement d'images.

2.3 Nos choix de modélisation

Lorsque l'utilisateur exprime son besoin, il doit se voir indiquer en réponse la séquence d'instructions à suivre pour réaliser son besoin. C'est ce but qui a guidé nos choix de modélisation des connaissances d'utilisation des traitements. Chaque mode d'emploi est composé d'étapes. Une même étape peut se réaliser de différentes façons suivant le contexte. Par exemple, «importer des données dans le SIG ou le programme considéré», se traduira par «appliquer tel traitement de conversion de format», puis «appliquer tel traitement de changement de projection», etc.

Comme dans MASK, plusieurs types de connaissances sont représentés dans notre modèle. Nous en avons introduit quatre. Les concepts (par exemple «distance euclidienne») et les modes d'emploi (spécifiques à un traitement – p.ex. «faire une requête topologique avec Geoconcept» –, ou génériques – p.ex. «interfacer du code Java et du C»–, «améliorer un Modèle Numérique de Terrain»[5]) sont des connaissances destinées à la lecture humaine. C'est-à-dire qu'elles sont manipulables informatiquement (elles font l'objet de requêtes et sont affichées), mais leur signification n'est pas exploitée par l'application; elles ne sont pas dotées de sémantique opérationnelle. Cela aurait été le cas si, pour reprendre nos exemples, le concept «distance euclidienne» avait été défini dans un langage formel qui aurait effectivement permis le calcul, ou si le mode d'emploi «convertir des données au format *shape* en *MIF*» avait été décrit dans un langage comme OWL-S, permettant l'invocation effective du service Web réalisant le changement de format [6].

Au contraire des concepts et modes d'emploi destinés à la lecture humaine, les règles d'adaptation et les heuristiques de choix de ces règles sont dans notre modèle des connaissances opérationnalisables (cf. tableau1). En effet elles sont utilisées par l'application pour déterminer les réalisations des étapes. Elles devraient également être intelligibles pour l'utilisateur lambda, mais les aspects procédural et déclaratif sont difficiles à concilier. En l'état actuel de nos travaux les règles d'adaptations sont consultables et saisissables sous forme de règles *Si contexte Alors Adaptation* (lien entre une étape et sa réalisation), mais les heuristiques ne sont accessibles qu'au développeur de l'application de gestion des métadonnées.

Les étapes des modes d'emploi n'étant pas toutes à mettre au même niveau, et ne se suivant pas toujours en séquence, il n'est pas satisfaisant de les présenter sous forme de liste plate. C'est pourquoi l'auteur d'un mode d'emploi est libre d'employer une numérotation hiérarchique arborescente (comme dans un livre on trouve chapitre 1, section 2.1, sous-section 2.2.3, etc.). Le *ou logique* permet d'indiquer les alternatives entre étape de même profondeur. Les structures de contrôles *for, while, split* (exécution simultanée), etc[7]

[5] Les MNT décrivent la forme et la position de la surface du sol. Ils comportent souvent des défauts, des artefacts qu'il est possible de corriger, par exemple en évitant qu'une rivière remonte ou qu'une crête soit plate.
[6] Précisons cependant à propos de la sémantique opérationnelle que les concepts sont définis dans une ontologie et organisés avec les relations de subsumption (distance Euclidienne et distance de Hausdorff sont deux spécialisations du concept de distance qui lui même est une spécialisation de concept mathématique, etc.)
[7] le détail des descripteurs de *processus* selon OWL-S peut être trouvé dans (The OWL Services Coalition, 2004).

utilisées par OWL-S pour l'agencement des services W eb ne nous ont pas paru utiles dans le cadre de nos objectifs. La notation du point précédent permet de contourner la difficulté de saisir en mode texte des graphes de type et/ou habituellement visualisés sous forme graphique. On touche là une différence notable de nature entre les diagrammes MASK ou UML et nos métadonnées avant tout textuelles. Ceci dit, il est toujours possible lors de la saisie d'inclure des images qui seront affichées à titre d'illustration (les références peuvent en effet pointer sur tout type de documents : images, URL, ou plus classiquement manuels dont on a souligné en 2.1 la complémentarité avec nos métadonnées).

Concernant les instructions, deux types ont été distingués, selon que l'utilisateur soit prêt à programmer ou simplement à utiliser un logiciel. Beaucoup de parties du code des programmes sont routinières; en informatique géographique c'est le cas des instructions qui permettent de faire des requêtes sur les bases de données géographiques, de charger des données dans les structures de données permettant de les manipuler, d'invoquer les méthodes des objets courants, etc. Cela explique l'intérêt de décrire des instructions types et les modèles de code (*code template*) qui leur correspondent.

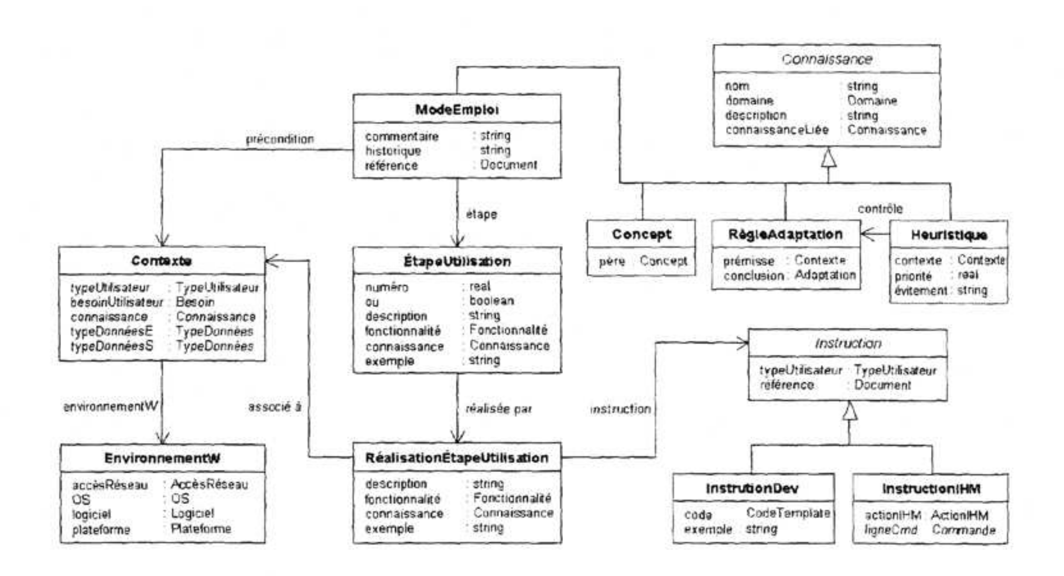


Figure 2-1 : Modélisation des modes d'emploi

3 Adapter les connaissances d'utilisation au contexte

3.1 Les règles d'adaptation des modes d'emploi

Les modes d'emploi doivent être adaptés aux contextes d'utilisation. Mais comme il n'est pas envisageable de stocker à l'avance les adaptations de tous les cas de figure possibles, il faut nécessairement mettre en place un mécanisme pour dériver l'information recherchée de celle présente dans la base de métadonnées, le contexte ayant été spécifié par l'utilisateur. Par exemple, si un traitement requiert des données *shape* et que l'utilisateur possède des données *MIF*, le serveur de métadonnées doit lui indiquer comment effectuer une conversion de format à l'aide des traitements adéquats. Le tableau 1 montre en pseudo-langage quelques règles à exprimer. Ces règles constituent une partie des connaissances à faire exprimer par les experts. Dans le cas présent, à la difficulté liée à l'aspect parfois tacite de ces connaissances, s'ajoute la contrainte de les acquérir sous une forme opérationnalisable. Notre réponse à ce double problème est la suivante.

Par la présence dans le formulaire de saisie d'un champ *Règle d'adaptation*, l'auteur de description de traitement est incité à songer aux éventuelles connaissances d'adaptations qu'il n'aurait pas spontanément formulées. Les règles d'adaptation déjà existantes, consultables, servent d'exemples. L'expressivité des règles repose grandement, dans la partie *prémisse*, sur la capacité du contexte à représenter la variété des cas de figure possibles. Il faudrait donc également rendre possible la saisie de nouveaux besoins particuliers susceptibles de caractériser le contexte. Ce travail est en cours, comme celui qui vise à opérationnaliser les connaissances heuristiques jusqu'à présent cantonnées à la simple forme de conseils à l'utilisateur.

<u>Conversion de format:</u> SI entrée_fournie..format = F1 AND entrée_attendue.format = F2 AND F1!= F2 ALORS nouvelle_étape (conversion(F1, F2))	<u>Logiciel indisponible :</u> SI unavailable (X) AND access.network = intranet ALORS nouvelle_étape (client Citrix) /* connection à une nouvelle machine */
<u>Interfaçage de langages:</u> SI contexte_utilisateur.langage = L1 AND traitement.langage = L2 AND L1!= L2 ALORS nouvelle_étape (interface(L1, L2))	<u>RAM insuffisante cause nbObjet</u> SI entrée_fournie.pte("nbObjets"). précondition.environnementW.RAM < RAM requise AND /* ... */ ALORS pbRAMinsuffisante

Tableau 1 : Exemples de règles d'adaptation des modes d'emploi au contexte de l'utilisateur

3.2 Quel formalisme pour stocker ces règles?

Les règles d'adaptation au contexte sont des métadonnées de traitements comme les autres. Elles sont donc stockées dans le même format, en l'occurrence XML. Notre modèle de métadonnées présenté plus haut (fig. 1 et 2) est en fait le *modèle conceptuel* (la forme du diagramme de classes étant un impératif du projet de recherche dans lequel s'inscrit notre travail) qui trouve sa traduction dans le *modèle d'implémentation* au format XML Schema. Pour mettre en oeuvre un moteur d'inférence plusieurs voies s'of fraient à nous. Comme (Gandon et Sadeh, 2003), générer du CLIPS avec une feuille XSLT pour utiliser le moteur d'inférence JESS était une solution. Nous avons plutôt choisi d'écrire une feuille XSLT générant elle-même les feuilles XSLT correspondant à la forme opérationnalisable des règles. Une telle feuille produit, par application sur un fichier XML contenant le contexte d'utilisation spécifié par l'utilisateur et sur la base de métadonnées[8], le mode d'emploi adapté attendu au format HTML.

Une solution à laquelle nous n'avons pensé est d'utiliser les mécanismes d'inférences associés au langage OWL. En effet si parfois les ontologies jouent le rôle de terminologies ou de modèles de connaissances hors de toute exploitation informatique, il arrive aussi – c'est le cas dans le domaine du Web Semantique – qu'elles soient destinées à faire l'objet d'inférences. C'est d'ailleurs parce qu'à cet égard les usages sont variables qu'OWL se décline en 3 sous-langages (OWL Lite, DL et Full (W3C, 2004)) selon que l'on cherche expressivité ou capacités d'inférence. Mais les types d'inférences permis de façon générique par OWL ne correspondent pas à nos besoins. Il s'agit principalement de la vérification de consistance (*est-ce qu'une classe peut avoir une instance ?*), de la classification (A est-elle une sous classe de B ?) et de la classification d'instance (*à quelle classe appartient un individu ?*) (Knublauch *et al.*, 2004). Ces inférences peuvent être ef fectuées de façon générique sur les ontologies OWL car elles reposent sur la sémantique formelle des relations unissant les concepts et les individus, sur l'existence de fonctions d'interprétation (Euzenat, 2004). Par exemple l'inférence qui permet de déduire que *«Corse et Ajaccio sont deux lieux apparentés puisque Napoléon est né dans les deux à la fois»* repose sur la fonction d'interprétation du mot clé cardinality, affecté en l'occurrence dans l'ontologie considérée de la valeur «1» pour la relation lieu_de_naissance. Contrairement à ce type d'exemple, les règles d'adaptation au contexte que nous voulons mettre en oeuvre s'appuient sur des relations spécifiques à notre modèle. Ces règles doivent donc être écrites de façon *ad hoc*.

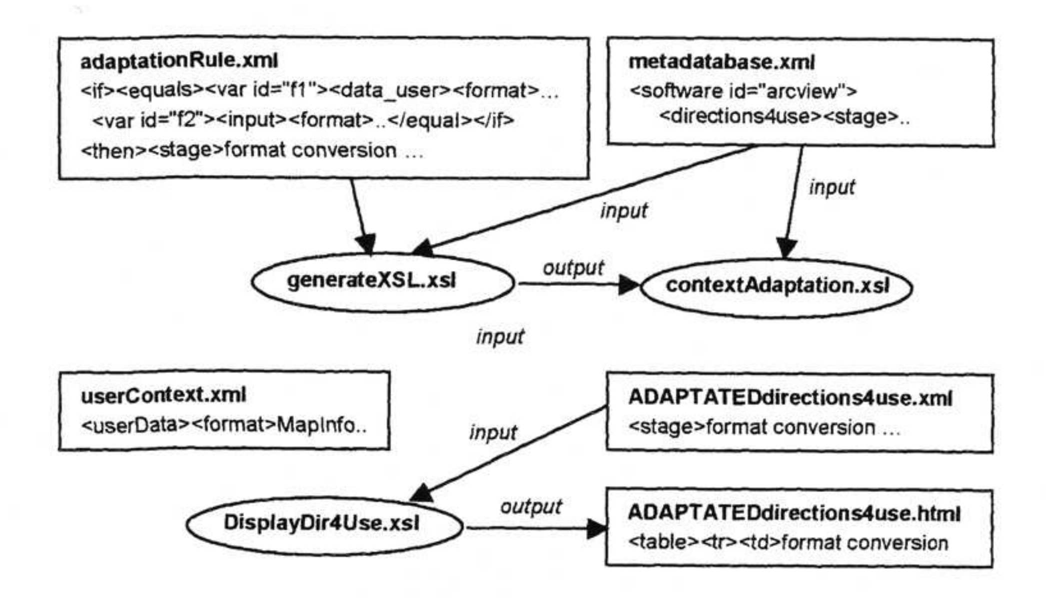


Figure 3-1 : Règles d'adaptation, de l'acquisition à l'exécution. Le lien entre generateXSL.xsl et contextAdapatation.xsl est le cœur du mécanisme; generateXSL.xsl génère dynamiquement les règles sous forme de *templates* XSLT

[8] L'expression XPath «document(' *file*')/*path*» permet d'accéder à plusieurs sources XML au cours d'une même passe XSLT.

4 Consulter les métadonnées, en saisir de nouvelles

4.1 Le serveur de métadonnées, l'interface utilisateur

Une base de métadonnées expérimentale et créée conformément au modèle défini ci-dessus est consultable et modifiable via un site intranet de l'IGN. Entre l'utilisateur du site équipé d'un navigateur Web et la base de métadonnées se trouve le serveur d'application qui effectue la reformulation des requêtes fournies, l'adaptation au contexte, la gestion de l'IHM, etc.; nous sommes dans le cadre d'une architecture 3 tiers classique.

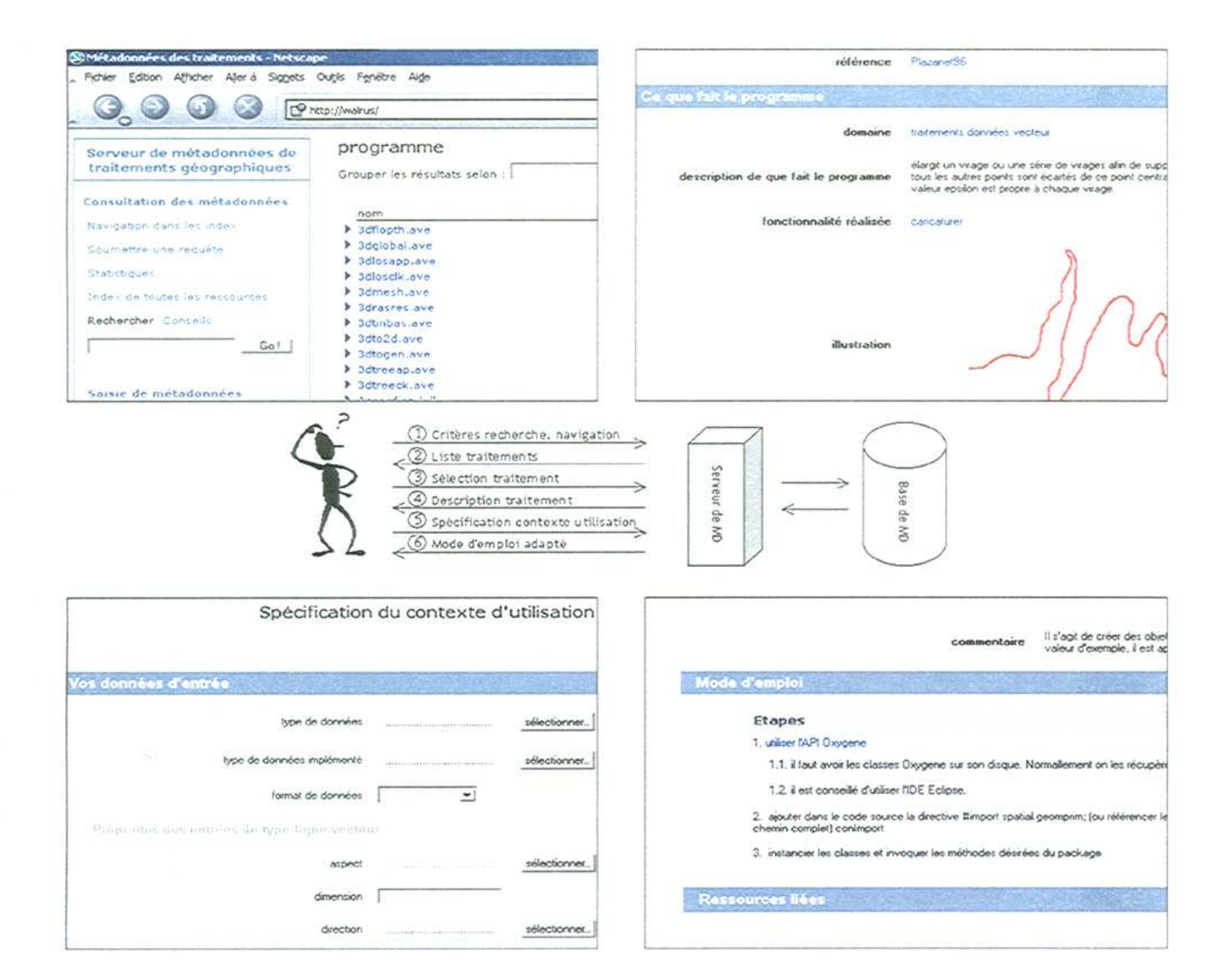


Figure 4-1 : Illustration d'un cas typique de consultation de la base de métadonnées

L'interface utilisateur propose les fonctionnalités traditionnelles de systèmes d'aide qui sont la recherche plein-texte de mots clés ou la navigation dans les index. A cela s'ajoute la possibilité de soumettre via des formulaires HTML des requêtes telles que *«quels sont les traitements dont l'auteur est Léon et la fonctionnalité réalisée caricature?»*. Nous gérons les relations de subsumption grâce à la génération dynamique d'index inversés. Le principe est simple : étant donné un ensemble d'index décrivant les relations unissant des entités, on construit de nouveaux index décrivant les relations inverses. Typiquement, partant de pages Web contenant des listes de mots, on construit les index inverses qui décrivent pour chaque mot les pages Web qui les contiennent. Ainsi, la requête précédente prendra en compte tous les traitements qui réalisent une sous-fonctionnalité de *caricaturer* (*amplification, dilatation*, etc.). Puisque nous avons automatisé l'export des classifications de notre base de métadonnées en OWL (cf. §4.2), un autre moyen que la construction d'index inversés pour retrouver les liens de subsumption aurait été l'utilisation d'une API permettant le requêtage OWL. Nous avons pour l'instant écarté cette possibilité pour des questions de performance. En effet coupler

un moteur OWL au moteur XSLT sur lequel repose le cœur de l'application était possible mais lourd (génération de fichiers intermédiaires, temps de *parsing* [9]) et nous avons donc pour l'instant choisi de tout faire en XSLT.

Les résultats des requêtes peuvent être triés selon le critère désiré. La présence de liens hypertextes vers les diverses métadonnées de la base est systématique, ce qui donne à la navigation une souplesse semblable à celle des Topic Maps ; Topic Maps dont les avantages et l'affichage via le générateur de pages Web Omnigator d'Ontopia [10] nous sont connus, mais dont la philosophie s'opposait aux principes de modélisation que nous souhaitions mettre en œuvre (Caussanel *et al.*, 2002)(XML Topic Maps, 2001).

4.2 Le contenu de la base de métadonnées

En son état actuel, la base de métadonnées est riche d'une centaine de descriptions de traitements (SIG et packages Java pour la manipulation de primitives géographiques principalement), mais aussi surtout de descriptions de fonctionnalités géographiques (200), de type de données (35 au niveau abstrait, c'est-à-dire non implémenté), de concepts et de modes d'emploi, descriptions dont les ensembles respectifs forment les classifications qui servent de base à l'indexation des traitements – indexation sémantique pourrait-on dire puisque le sens des ressources est fixé et normalisé dans le cadre de travail – mais de façon générale ces descriptions sont utiles pour qui veut avoir un aperçu synthétique du domaine (fig.4)

Sans préjuger de la valeur de nos classifications, de toute façon appelées à être enrichies et éventuellement révisées au fil du temps [11], les rendre disponibles sous forme d'ontologie OWL pourrait constituer une participation au projet d'interopérabilité des services Web géographiques et plus généralement au partage de la connaissance. Nous-même aurions été heureux d'avoir accès sous forme d'ontologie à la classification des fonctionnalités selon une partie de la communauté géographique (ce que nous n'avons pu trouver que sous forme de parutions écrites). C'est pourquoi nous avons créé des feuilles XSLT exportant les classifications de notre base de données vers des ontologies en OWL. La figure 4 montre ainsi la hiérarchie des fonctionnalités avec l'éditeur Protégé 3.0

4.3 Semi-automatisation de l'acquisition des descriptions de programmes

Afin d'automatiser une partie de la saisie des formulaires de description des classes et méthodes Java, nous générons des scripts clients Javascript avec les informations fournies par le doclet que nous développé. Proposée par Sun, cette techniqueconsisteà créer une classe qui étend com.sun.javadoc.Doclet en redéfinissant la méthode start() et en modifiant ainsi le comportement standard de la commande javadoc, qui est de produire la classique documentation au format Html. Les informations du code à documenter, auxquelles on accède via l'API Reflection, étant normalement déjà indexées dans la base de métadonnées, l«indexation sémantique» automatique est en partie possible. Par exemple si le doclet indique qu'une variable est de type connu GM_Object, alors on peut en déduire qu'au niveau abstrait il s'agit d'une donnée *vecteur.*

Nous avons de plus, mais avec un succès mitigé, tenté de détecter les fonctionnalités par le comptage des lexèmes [12] communs aux commentaires de code et aux descriptions de la base de métadonnées. Pour extraire les commentaires et signatures de fonctions des autres langages que Java nous avons développé des parseurs basés sur des expressions régulières.

Conclusion

Si, en leur état actuel, le modèle de métadonnées défini et l'application développée permettent bien de répondre à la majorité des requêtes simples que l'analyse des besoins a fait apparaître, il reste néanmoins des cas où l'utilisateur ne peut être renseigné de façon satisfaisante, comparé à l'information que pourrait fournir un expert humain. En rendant opérationnalisable une partie des connaissances d'utilisation des traitements, nous avons fait un pas vers un système d'aide «intelligent». Que ce soit pour l'aide au paramétrage, la modélisation des connaissances générales du domaine géographique, ou l'aide à la programmation, les pistes ne manquent pas pour continuer dans cette voie. Pour de tels besoins, nous retenons de l'expérience de notre travail qu'avant même l'apport de mécanismes d'inférences simulant l'expert, c'est dans le modèle de métadonnées que réside «l'intelligence» puisqu'il permet la structuration et la transmission des connaissances. En cela nous nous situons dans la lignée des modèles de gestion de connaissances que nous avons étudiés.

---

[9] Lors d'une transformation XSLT le fichier XML source est chargé en mémoire vive sous la forme d'un DOM (Document Object Model) XML. Un DOM XML prend en mémoire vive environ 10 fois la taille du fichier XML source, ce qui pose des problèmes de performance à partir d'un certain seuil

[10] http://www.ontopia.net/omnigator/models/index.jsp

[11] Le versionnement des ontologies OWL est prévu (The OWL Services Coalition, 2004, §6) mais cela ne résout pas tous les problèmes causés par les révisions dans les indexations devenues obsolètes. La description des évolutions et les mises à jours seraient des pistes à creuser.

[12] on lemmatise les termes à l'aide de l'outil gratuit Unitex (http://www-igm.univ-mlv .fr/~unitex/)

Références

Bandza C., *Des méthodes de formalisation des connaissances et de MKSM en particulier* , thèse professionnelle du mastère MSIT, Management des Systèmes d'Informations et des Technologies, HEC-Mines, http://www.hec.ensmp.fr/ Theses/ Theses2000/Bandza.doc, 2000.

Bard S., *Méthode d'évaluation de la qualité de données géographiques généralisées Application aux données urbaines.* Thèse de doctorat, université Paris 6, 2004.

Bézard J-M. et Ariès S., *La méthode MASK - Présentation pour la capitalisation des connaissances* , http://perso.wanadoo.fr/ serge.aries/presentation/MASKmet/frame.htm, 2003

Caussanel J., Cahier J-P., Zacklad M., Charlet J., *Les Topic Maps sont-ils un bon candidat pour l'ingénierie du Web Sémantique ?*, In actes de la conférence Ingénierie des connaissances, Rouen, 2002.

Clouard R., Elmoataz A. et Revenu M., *Une modélisation explicite et opérationnelle de la connaissance en Traitement d'Images*, actes de la conférence RFIA'98, Clermont-Ferrand, 1998.

Ermine J-L., *La Gestion des connaissances*, Hermès Science Publication (diff. Lavoisier), 2003. *Voir aussi : J.-L. Ermine, Introduction à la méthode MASK, http://perso.wanadoo.fr/serge.aries/presentation/MASKint/frame.htm.*

Euzenat J., *Chouette alors ! Un langage d'ontologies pour le web.* Conférence invitée d'IC'2004, 15 èmes journées francophones d'ingénierie des connaissances, Lyon, 2004.

Gandon F. et Sadeh N. *Gestion de connaissances personnelles et contextuelles, et respect de la vie privée* . In actes d'IC'2004, 15èmes journées francophones d'ingéniérie des connaissances, Lyon, 2004.

Knublauch H., Musen M.A. et Noy N.F., *Creating Semantic Web (OWL) Ontologies with Protégé.* Stanford Medical Informatics, http://protege.stanford.edu/ plugins/owl/publications/2004-07-06-OWL-T ut%orial.ppt, 2004.

The OWL Services Coalition. OWL-S: *Semantic Markup for Web Services.* http://www.daml.org/services/owl-s/1.0/owl-s.html, 2004.

XML Topic Maps 1.0, *TopicMaps.org Specification*,. http://www.topicmaps.org/xtm /1.0, 2001